# Xspeak: A Use for a Speech Interface in a Windowing System

Mark S. Ackerman, Sanjay Manandhar and Chris M. Schmandt

Media Laboratory, M.I.T.
Cambridge, MA 02139 USA
(617) 253-5156

We discuss an application to add speech recognition capabilities to a window system in order to navigate among the windows of most immediate use to the user. *Xspeak* is a system designed to manipulate the windows in the X Window System using voice input. We added speech not as a replacement for the keyboard and mouse, but as a supplemental mode of communication in order to make the user interface more convenient and intuitive. Speech input not only controlled the layout and focus of the windows (tasks which are normally relegated to a window manager), but it also enabled the user to create and activate windows for particular tasks such as reading mail and editing text.

## Introduction

With the growing popularity of window systems on computer workstations, there is a need for a suitable interface to window management. To cope with limited screen real-estate, many window systems allow windows to overlap. Since overlapping windows may obscure other windows, a mechanism to find and navigate among them is required. Normally the combination of mouse and keyboard is used to manipulate windows.

The use of keyboard and mouse poses several problems, however. First, although the screen is two dimensional, the overlap of the windows provides an additional dimensionality of depth. If many windows exist, it may be difficult to find any particular window. Second, using the mouse requires the user to move his hand from the keyboard. For example, depending on the window manager, sometimes the desired action is executed by pressing a tiny visual button or by clicking on the window's titlebar. Speech has neither of these drawbacks.

In order to make the user interface more convenient and intuitive, we added speech as a supplemental mode of communication in Xspeak. This added input channel eased the need for accurate manipulation of mouse and keyboard to control the configuration of windows. Xspeak solved the problem of finding stacked windows when they are buried.

## Xspeak

Xspeak is a standard X Window System ("X") application and, as such, it is transparent to other X applications. Since X is a client-server system, Xspeak interacts through the X server with the various applications and with the window manager, a specialized X application that controls the user interface for moving, restacking, and resizing windows. [1] We did not modify the X Window System, any window manager, or any application in order to accommodate Xspeak.

[1] There are many window managers available under X, and they can be interchanged by the user.

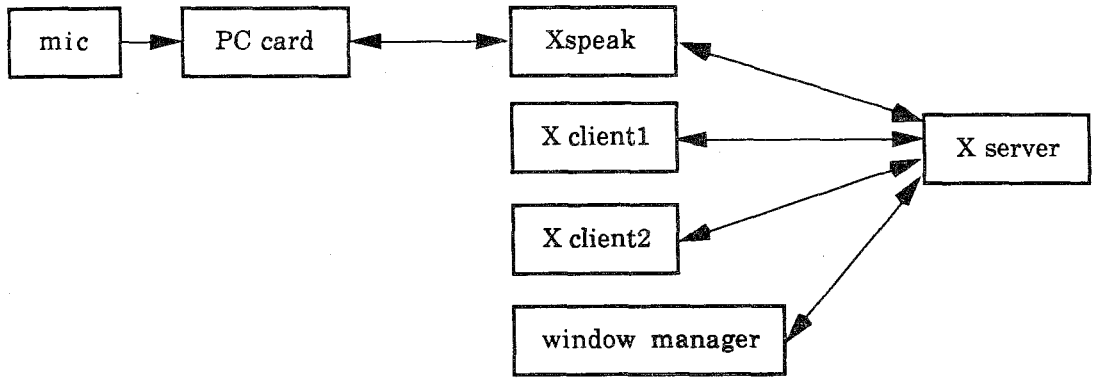


Figure 1: System diagram of Xspeak

One may think of Xspeak as logically consisting of two modules: a speech module to handle recognition and a window system module to interact with the window system. (See Figure 1.) Xspeak runs on the local workstation; Xspeak uses a Texas Instrument speech card for recognition. The speech card resides in a PC which communicates with the X workstation over a serial line. We used a Sennheiser M-80 super-cardioid microphone. Although most studies use head-mounted, noise-cancelling microphones, we felt that they were not comfortable enough for everyday use.

Xspeak's speech module interacts with both the PC card and the user. Xspeak, for example, needs to solicit status information on success or failure of recognition from the recognition card, update the vocabulary list, and perform disk I/O on the PC hard disk. The speech module is also responsible for the Xspeak control panel that provides the user with visual feedback on whether the word was recognized, visual buttons to invoke the training of the vocabulary templates, and to quit the program.

The window system module makes the right requests to the X server to effect the desired action. Xspeak works by associating windows with voice names in the speech recognizer's vocabulary. The result of the recognition (i.e., whether a word was recognized or not) is reported to Xspeak. When any utterance matches the already trained set of vocabulary words (i.e., it is recognized), Xspeak identifies the corresponding window and takes the appropriate action.

In most cases, Xspeak requests that the X Window System bring the window to the top of the window stack. The window system module also moves the mouse pointer to the middle of the raised window. If the window does not currently exist, a window with the appropriate application is created as specified in a user-defined configuration file. In addition, the user can add a new name for a window by clicking on it and training a new template in the speech recognizer. It is also possible to completely lower or raise windows. Thus, users can navigate among windows and rearrange them through speech.

Internally, there is a mapping between a vocabulary template and the window ID through the window name resource, a property of the top-level window. A dynamically trained word is mapped

to its corresponding window through only the unique window ID that the X server provides.

To study the effectiveness of speech input, we gave Xspeak to four students and two of the authors for periods ranging from two weeks to two months. All used Xspeak to navigate around in X while developing software as part of their daily work. All these users were familiar with the X Window System, and only one of the authors had used speech recognition. For the findings, see [1].

The study led to Xspeak II, which is currently under development. Xspeak II will include a more full-featured language to allow window navigation, conditional processing, and interaction with direct manipulation controls. In addition, to increase recognition accuracy and allow larger vocabularies, application driven vocabulary subsetting will be employed. These features are described in [2].

## Use of Voice in a Windowing System

In general, users need to provide 4 different kinds of input in a windowing system.

1. *Application data.* This is the text input for a word processor or the numeric data in a spreadsheet.
2. *Control data.* This input changes the application state. You might change the state of a word processor, for example, by command sequences or by clicking on a direct manipulation object, such as visual button or scrollbar.
3. *Navigation.* This input allows moving among applications or changing input focus. In a windowing system, for example, you might move the mouse between two windows.
4. *Layout.* In a windowing system, the user often wishes to re-layout the visual appearance of his screen by changing the size or the position of some windows.

Some systems are moving toward providing ways to layout not only windows, but groups of windows into tasks; this could be considered a fifth type of input in the system. Note that these categories in any given system are not necessarily distinct; we separate them here for analytical purposes.

The interesting question, then, is what the corresponding input methods for data might be in either an auditory system or in a mixed media system. Having different input media available means that the user can select the "style" most appropriate to him and to the functionality.

Xspeak is an exploration in this mixed media space, having text and graphics output and speech input. It explores the area defined by requirement 3, navigation, and requirement 1, application data. Our continuing work with Xspeak II is addressing requirement 2, interaction with the direct manipulation controls in the interface.

The current version of Xspeak directly addresses the question of inter-application navigation. By allowing users to easily move among applications through voice, it is easy for the user to change his current application. Through this mechanism, Xspeak can raise or lower the appropriate window; the separation of client from server in X allows any client to control the windows of another client through the server.

In a mixed media system, one would still want to use the keyboard for application input, primarily for its speed. Under X, sharing the keyboard among applications is straightforward. As the user navigates among his windows, most window managers in X automatically reassign keyboard focus to the appropriate window to send application input to the proper application.

Interacting with the direct manipulation controls is less straightforward. Since the X Toolkit (Xt) controls exist within the client, and because there are no "hooks" for sending messages to the controls themselves, it is more awkward to interact with those controls. Xspeak II can send artificial mouse events to the control's window, but this requires knowing the window IDs for each control. In a windowing system built for both voice and visual interaction, one would like either the controls to be accessible via a toolkit server or some other way to easily identify controls from another application.

Allowing the user to change his mixed media layout is most challenging. In a visual environment, a user moves around windows to allow different views of his visual (and therefore, task) space. In a visual system, the user gains by having different windows, and he determines screen real-estate for his windows to maximize important output. Xspeak II will be able to address visual layout to some extent.

In a mixed media environment, what layout entails is less clear. Ultimately, we would like the user to be able to arrange different media differently, placing audio in an audio space, windows in a visual space, and the interaction in some mutual space. It is clear that windows allow users to process their tasks in parallel. Similarly, speech and audio are semantically rich, but computationally difficult. In a sense, the audio equivalent of a visual window is required. One such possibility is to fuse the visual windowing system with an audio system; each can reinforce the other. Explorations are required to handle speech and audio in ways that support multiple, parallel outputs, with the corresponding multiple input methods.

## Acknowledgments

Debby Hindus performed the majority of the user evaluation for Xspeak. The authors would also like to thank Wendy Mackay, Gale Martin, Ralph Swick, and Dan Swinehart for their insightful comments and suggestions. This project was funded by the MIT X Consortium and Sun Microsystems, Inc.

## References

[1] Chris Schmandt, Mark S. Ackerman, and Debby Hindus. Augmenting a window manager with speech input. *IEEE Computer*, August 1990. Forthcoming.

[2] Chris Schmandt, Debby Hindus, Mark S. Ackerman and Sanjay Manandhar. Observations on using speech input for window navigation. *Proceedings of Interact90, 3rd IFIP Conference on Human-Computer Interaction, IFIP*. Forthcoming.